

NNN		NNN	MMM	MMM	LLL
NNN		NNN	MMM	MMM	LLL
NNN		NNN	MMM	MMM	LLL
NNN		NNN	MMMMMM	MMMMMM	LLL
NNN		NNN	MMMMMM	MMMMMM	LLL
NNN		NNN	MMMMMM	MMMMMM	LLL
NNNNNN		NNN	MMM	MMM	LLL
NNNNNN		NNN	MMM	MMM	LLL
NNNNNN		NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNNNNN	NNN	MMM	MMM	LLL
NNN	NNNNNN	NNN	MMM	MMM	LLL
NNN	NNNNNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLL
NNN	NNN	NNN	MMM	MMM	LLLLLLLLLLLLLLLL
NNN	NNN	NNN	MMM	MMM	LLLLLLLLLLLLLLLL
NNN	NNN	NNN	MMM	MMM	LLLLLLLLLLLLLLLL

\_S

Ps

NP

NP

SG

SO

NP

PA

\_L

NN	NN	MM	MM	LL	EEEEEEEEEE	NN	NN	TTTTTTTTTT	RRRRRRRR	YY	YY
NN	NN	MM	MM	LL	EEEEEEEEEE	NN	NN	TTTTTTTTTT	RRRRRRRR	YY	YY
NN	NN	MMM	MMM	LL	EE	NN	NN	TT	RR	YY	YY
NN	NN	MMM	MMM	LL	EE	NN	NN	TT	RR	YY	YY
NNNN	NN	MM	MM	LL	EE	NNNN	NN	TT	RR	YY	YY
NNNN	NN	MM	MM	LL	EE	NNNN	NN	TT	RR	YY	YY
NN	NN	MM	MM	LL	EEEEEEEEEE	NN	NN	TT	RRRRRRRR	YY	YY
NN	NN	MM	MM	LL	EEEEEEEEEE	NN	NN	TT	RRRRRRRR	YY	YY
NN	NNNN	MM	MM	LL	EE	NN	NNNN	TT	RR	YY	YY
NN	NNNN	MM	MM	LL	EE	NN	NNNN	TT	RR	YY	YY
NN	NN	MM	MM	LL	EE	NN	NN	TT	RR	YY	YY
NN	NN	MM	MM	LL	EE	NN	NN	TT	RR	YY	YY
NN	NN	MM	MM	LLLLLLLLLL	EEEEEEEEEEEE	NN	NN	TT	RR	YY	YY
NN	NN	MM	MM	LLLLLLLLLL	EEEEEEEEEEEE	NN	NN	TT	RR	YY	YY

```

LL          IIIII
LL          IIIII
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LL          II
LLLLLLLLLLL IIIII
LLLLLLLLLLL IIIII
SSSSSSSSS
SSSSSSSSS
SS
SS
SS
SS
SSSSSS
SSSSSS
SS
SS
SS
SS
SSSSSSSSS
SSSSSSSSS

```

```
0001 0 XTITLE 'Network Management Listener entry point'
0002 0 MODULE NMLENTRY (IDENT = 'V04-000',
0003 0 ADDRESSING_MODE (NONEXTERNAL=GENERAL),
0004 0 ADDRESSING_MODE (EXTERNAL=GENERAL)) =
0005 1 BEGIN
0006 1
0007 1 *****
0008 1 *
0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0011 1 * ALL RIGHTS RESERVED.
0012 1 *
0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0018 1 * TRANSFERRED.
0019 1 *
0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0022 1 * CORPORATION.
0023 1 *
0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0026 1 *
0027 1 *****
0028 1
0029 1
0030 1
0031 1 ++
0032 1 FACILITY: DECnet-VAX V2.0 Network Management Listener
0033 1
0034 1 ABSTRACT:
0035 1
0036 1 This module contains the entry points for the
0037 1 callable interface for the NML sharable image.
0038 1
0039 1 ENVIRONMENT: VAX/VMS Operating System
0040 1
0041 1 AUTHOR: Tim Halvorsen, July 1981
0042 1
0043 1 MODIFIED BY:
0044 1
0045 1 V03-006 MKP0007 Kathy Perko 4-Aug-1983
0046 1 Add support for faster node permanent database.
0047 1
0048 1 V03-005 MKP0006 Kathy Perko 20-April-1983
0049 1 Add support to call MOM for service functions.
0050 1
0051 1 V03-004 MKP0005 Kathy Perko 9-Nov-1982
0052 1 Consolidate two routines that validate the Network
0053 1 Management versions for NML and NCP. Also,
0054 1 update to version 4.0.0.
0055 1 Add logging of NICE messages to NML$WATCHER
0056 1 to keep a running log of all NICE messages handled on
0057 1 a node for as long as watcher is defined.
```

58	0058	1	!			
59	0059	1	!	V03-003	MKP0004	Kathy Perko 18-Oct-1982
60	0060	1	!			Change NML so any permanent database files left open
61	0061	1	!			when a command has been processed are closed.
62	0062	1	!			
63	0063	1	!	V03-002	MKP0003	Kathy Perko 8-Sept-1982
64	0064	1	!			Move assign for NETACP QIO channel to NML\$NETQIO. This
65	0065	1	!			allows NML to process NCP commands to the permanent data
66	0066	1	!			base even if NETACP is not mounted.
67	0067	1	!			
68	0068	1	!	V03-001	MKP0002	Kathy Perko 16-June-1982
69	0069	1	!			Change some global names to make them more meaningful.
70	0070	1	!			
71	0071	1	!	V02-002	MKP0001	Kathy Perko 04-Feb-1982
72	0072	1	!			Allow NCPs with version numbers greater than or equal
73	0073	1	!			to 3.0 (as well as 2.0) to talk to this NML.
74	0074	1	!			
75	0075	1	!	V001	TMH0001	Tim Halvorsen 12-Oct-1981
76	0076	1	!			Change argument to NML\$INITIALIZE to accept the
77	0077	1	!			version number of NICE to be spoken, rather than the phase.
78	0078	1	!			Remove obsolete comment.
79	0079	1	!--			

```
.. 81      0080 1  %SBTTL 'Declarations'
.. 82      0081 1
.. 83      0082 1
.. 84      0083 1  !! TABLE OF CONTENTS:
.. 85      0084 1  !!
.. 86      0085 1
.. 87      0086 1  FORWARD ROUTINE
.. 88      0087 1      NML$INITIALIZE,      ! Initialize NML
.. 89      0088 1      NML$PROCESS_NICE:    NOVALUE,      ! Process a NICE message
.. 90      0089 1      NML$TERMINATE:      NOVALUE,      ! Terminate NML
.. 91      0090 1      NML INITLOG:        NOVALUE,      ! Initialize message logging
.. 92      0091 1      NML$SEND,          ! Send response to caller
.. 93      0092 1      NML$LOOP2:         NOVALUE,      ! Phase II passive loopback
.. 94      0093 1      NML$PHASE2:        NOVALUE,      ! Phase II NICE processing
.. 95      0094 1      NML$MAINHANDLER;    ! Main condition handler
.. 96      0095 1
.. 97      0096 1
.. 98      0097 1  !! INCLUDE FILES:
.. 99      0098 1  !!
100      0099 1
101      0100 1  LIBRARY 'LIB$:NMLLIB';      ! Facility-wide definitions
102      0101 1
103      0102 1  LIBRARY 'SHRLIB$:NMLIBRY';  ! NICE definitions
104      0103 1
105      0104 1  LIBRARY 'SYSS$LIBRARY:STARLET'; ! VMS common definitions
106      0105 1
107      0106 1  !!
108      0107 1  !! OWN STORAGE:
109      0108 1  !!
110      0109 1
111      0110 1  OWN
112      0111 1      nml$gl_response_rtn,    ! Address of response action routine
113      0112 1
114      0113 1      nml$b_ph2link: BYTE INITIAL(false), ! Phase II link flag (true->connected)
115      0114 1      nml$w_nicechan: WORD;    ! Phase II channel of NICE object
116      0115 1
117      0116 1  !!
118      0117 1  !! EXTERNAL REFERENCES:
119      0118 1  !!
120      0119 1
121      0120 1  $NML_EXTDEF;                ! Define common external data
122      0121 1
123      0122 1  EXTERNAL
124      0123 1      nml$gq_proprvmsk:  BBLOCK [8],
125      0124 1      nml$gb_ncp_version: VECTOR [3,BYTE],    ! NICE version being spoken
126      0125 1      nml$gl_logmask,
127      0126 1      nml$gw_watcher_chan: WORD,
128      0127 1      nml$gq_watcher_dsc;
129      0128 1
130      0129 1  EXTERNAL ROUTINE
131      0130 1      lib$asn_wth_mbx,
132      0131 1      nml$closefile,
133      0132 1      nml$change,
134      0133 1      nml$sv2_compatibility,
135      0134 1      nml$debug_msg,
136      0135 1      nml$error-1,
137      0136 1      nml$logaltpdb,
```

NMLENTRY  
V04-000

Network Management Listener entry point  
Declarations

H 9  
15-Sep-1984 23:58:02  
14-Sep-1984 12:50:08

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[NML.SRC]NMLENTRY.B32;1 Page 4 (2)

:	138	0137	1	nml\$parse_init,
:	139	0138	1	nml\$read,
:	140	0139	1	nml\$call_mom,
:	141	0140	1	nml\$trnlognum,
:	142	0141	1	nml\$zero;

NMLENTRY  
V04-000

```
144 0142 1 XSBTTL 'NML$INITIALIZE Initialization routine'
145 0143 1
146 0144 1 GLOBAL ROUTINE NML$INITIALIZE (VERSION) =
147 0145 1
148 0146 1 !++
149 0147 1 This is the initialization routine for the DECnet-VAX Network
150 0148 1 Management Listener. This module initializes the own storage
151 0149 1 in preparation for processing NICE messages. It also validates
152 0150 1 the Network Management Version of NICE that the caller (NCP or
153 0151 1 whoever) is using to talk to NML. If it is a version that this
154 0152 1 version of NML does not allow, return a version mismatch.
155 0153 1
156 0154 1 Inputs:
157 0155 1 version = Address of 3 byte version number of NICE to be spoken.
158 0156 1 1.3.0 = NICE V1.3.0 (Phase II)
159 0157 1 2.0.0 = NICE V2.0.0 (Phase III)
160 0158 1 3.0.0 = NICE V3.0.0 (Phase III with multipoint)
161 0159 1 4.0.0 = NICE V4.0.0 (Phase IV) - default
162 0160 1
163 0161 1 Implicit outputs:
164 0162 1 nml$gb_cmd_ver Indicates which tables to use when parsing the
165 0163 1 NICE message.
166 0164 1
167 0165 1 Outputs:
168 0166 1 Returns SS$ BADPARAM (Bad parameter) if there is a version mismatch.
169 0167 1 NML$GQ_PROPRVMSK = Current privilege mask
170 0168 1 NML$GB_NCP_VERSION = NICE version number
171 0169 1 --
172 0170 1
173 0171 2 BEGIN
174 0172 2
175 0173 2 BUILTIN
176 0174 2 NULLPARAMETER;
177 0175 2
178 0176 2 OWN
179 0177 2 GETPRVLST : BLOCK [7] ! Argument block for $GETJPI
180 0178 2 INITIAL (WORD (8, JPI$ PROCPRIV),
181 0179 2 NML$GQ_PROPRVMSK,
182 0180 2 0,
183 0181 2 0);
184 0182 2
185 0183 2 !
186 0184 2 Store version number of NICE being spoken from now on. Only major
187 0185 2 version numbers are distinguished.
188 0186 2
189 0187 2
190 0188 2 IF NULLPARAMETER(1) ! If no parameter specified,
191 0189 2 THEN
192 0190 2 BEGIN
193 0191 2 CH$MOVE(3, nml$ab_nml_nmv, ! then default to current version
194 0192 2 nml$gb_ncp_version);
195 0193 2 nml$gb_cmd_ver = nml$sc_phase3_or_4; ! Use Phase III and IV NICE parsing tables
196 0194 2 END
197 0195 2 ELSE
198 0196 2
199 0197 2 !
200 0198 2 ! Validate the three byte version number supplied by the process attempting
```

```
201 0199 2 ! to connect with NML.
202 0200
203 0201 BEGIN
204 0202 IF CH$RCHAR(.version) EQL 2 ! Allow V2.0.0
205 0203 OR CH$RCHAR (.version) EQL 3 ! or allow V3.0.0
206 0204 OR CH$GEQ(3, .version, ! or current version (4.0) or higher.
207 0205 3, nml$ab_nml_nmv, 0) THEN
208 0206 BEGIN
209 0207 CH$MOVE(3, .version, ! Use specified (and validated) version
210 0208 nml$gb_ncp_version);
211 0209 nml$gb_cmd_ver = nml$c_phase3_or_4; ! Use Phase III and IV NICE parsing tables
212 0210 END
213 0211 ELSE
214 0212 IF CH$RCHAR(.version) LSSU 2 THEN ! If less than V2.0.0 NICE,
215 0213 nml$gb_cmd_ver = nml$c_phase2 ! Then mark Phase II
216 0214 ELSE
217 0215 RETURN ss$_badparam; ! Signal invalid NICE version #
218 0216 END;
219 0217
220 0218 ! Get process privilege mask.
221 0219
222 0220 $GETJPI (ITMLST = getprvlst);
223 0221
224 0222 ! Initialize logging.
225 0223
226 0224
227 0225 nml initlog ();
228 0226 RETURN ss$_normal;
229 0227 1 END;
```

.TITLE NMLENTRY Network Management Listener entry poi  
nt

.IDENT \V04-000\

.PSECT \$OWNS,NOEXE,2

00000 NML\$GL\_RESPONSE\_RTN:

.BLKB 4

00 00004 NML\$B\_PH2LINK:

.BYTE 0

00005 .BLKB 1

00006 NML\$W\_NICECHAN:

.BLKB 2

0204 0008 00008 GETPRVLS:

.WORD 8, 516

00000000 00000000G 0000C .ADDRESS NML\$GQ\_PROPRVMSK

.LONG 0, 0

00018 .BLKB 12

.EXTRN NML\$GB\_EVTSRCTYP

.EXTRN NML\$GQ\_EVTSRCDSC

.EXTRN NML\$GW\_EVTCLASS

.EXTRN NML\$GB\_EVTMSKTYP

.EXTRN NML\$GQ\_EVTMSKDSC

.EXTRN NML\$GW\_EVTSNKADR

.EXTRN NML\$GW\_ACP\_CHAN

```

.EXTRN NML$GL_LOGMASK, NML$GQ_ENTSTRDSC
.EXTRN NML$AB_QIOBUFFER
.EXTRN NML$GQ_QIOBF DSC
.EXTRN NML$AB_EXEBUFFER
.EXTRN NML$GL_EXEDATPTR
.EXTRN NML$GQ_EXEDATDSC
.EXTRN NML$GQ_EXEBF DSC
.EXTRN NML$AB_RCVBUFFER
.EXTRN NML$GQ_RCVBF DSC
.EXTRN NML$AB_SNDBUFFER
.EXTRN NML$GQ_SNDBF DSC
.EXTRN NML$GL_RCV DATLEN
.EXTRN NML$AB_CPTABLE, NML$AB_MSGBLOCK
.EXTRN NML$AB_ENTITY_ID
.EXTRN NML$AB_QUALIFIER_ID
.EXTRN NML$AB_ENTITYDATA
.EXTRN NML$AB_NML NMV, NML$AB_PRMSEM
.EXTRN NML$AB_RECBUF, NML$AL_ENTINFTAB
.EXTRN NML$AL_PERMINFTAB
.EXTRN NML$AW_PRM DES, NML$GB_CMD_VER
.EXTRN NML$GB_ENTITY_CODE
.EXTRN NML$GB_ENTITY_FORMAT
.EXTRN NML$GL_QUALIFIER_PST
.EXTRN NML$GB_QUALIFIER_FORMAT
.EXTRN NML$GB_FUNCTION
.EXTRN NML$GB_INFO, NML$GB_OPTIONS
.EXTRN NML$GL_PRM CODE, NML$GL_PRS_FLGS
.EXTRN NML$GL_NML_ENTITY
.EXTRN NML$GQ_NETNAM DSC
.EXTRN NML$GQ_REC BUF DSC
.EXTRN NML$GW_PRMDESCNT
.EXTRN NML$GQ_PROPRVMSK
.EXTRN NML$GB_NCP VERSION
.EXTRN NPASGL_LOGMASK, NML$GW_WATCHER_CHAN
.EXTRN NML$GQ_WATCHER DSC
.EXTRN LIB$ASN WTH MBX
.EXTRN NML$CLOSEFICE, NML$CHANGE
.EXTRN NML$V2 COMPATIBILITY
.EXTRN NML$DEBUG MSG, NML$ERROR_1
.EXTRN NML$LOCAL PDB, NML$PARSE_INIT
.EXTRN NML$READ, NML$CALL MOM
.EXTRN NML$TRN LOGNUM, NML$ZERO
.EXTRN SYSSGETJPI

```

.PSECT SCODES,NOWRT,2

Address	Hex	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
---------	-----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

NMLENTRY  
V04-000

Network Management Listener entry point  
NMLSINITIALIZE Initialization routine

15-Sep-1984 23:58:02  
14-Sep-1984 12:50:08

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[NML.SRC]NMLENTRY.B32;1 Page 8  
(3)

64	18	04	BC	03	04	0D 13 0002B	BEQL 3\$		
						BC 91 0002D	CMPB @VERSION, #3	:	0203
						07 13 00031	BEQL 3\$	:	
						03 29 00033	CMPC3 #3, @VERSION, NMLSAB_NML_NMV	:	0204
						0B 1F 00038	BLSSU 5\$	:	
						BC F0 0003A	INSV @VERSION, #0, #24, NMLSGB_NCP_VERSION	:	0207
						02 90 00040	MOVB #2, NMLSGB_CMD_VER	:	0209
						0F 11 00043	BRB 7\$	:	0202
						BC 91 00045	CMPB @VERSION, #2	:	0212
						05 1E 00049	BGEQU 6\$	:	
						01 90 0004B	MOVB #1, NMLSGB_CMD_VER	:	0213
						04 11 0004E	BRB 7\$	:	
						14 D0 00050	MOVL #20, R0	:	0215
						04 00053	RET	:	
						7E 7C 00054	CLRQ -(SP)	:	0220
						7E D4 00056	CLRL -(SP)	:	
						00 9F 00058	PUSHAB GETPRVLST	:	
						7E 7C 0005E	CLRQ -(SP)	:	
						7E D4 00060	CLRL -(SP)	:	
						07 FB 00062	CALLS #7, SYS\$GETJPI	:	
						00 FB 00069	CALLS #0, NML_INITLOG	:	0225
						01 D0 00070	MOVL #1, R0	:	0226
						04 00073	RET	:	0227

; Routine Size: 116 bytes, Routine Base: \$CODE\$ + 0000

NM  
VO

```
0228 1 %SBTTL 'NML$PROCESS_NICE      Main command processing routine'
0229 1
0230 1 GLOBAL ROUTINE NML$PROCESS_NICE (msg_desc, resp_rtn): NOVALUE =
0231 1
0232 1 !++
0233 1      This routine is the main command processing routine.  NICE messages
0234 1      are parsed to determine the requested function and then the proper
0235 1      routine is called to perform the function.
0236 1
0237 1      Inputs:
0238 1
0239 1      msg_desc = Address of descriptor of NICE message
0240 1      resp_rtn = Address of action routine to call with NICE response
0241 1      The action routine is called with the following arguments:
0242 1          1) Address of descriptor of NICE response
0243 1
0244 1      Outputs:
0245 1
0246 1      None - control is returned after the last response has been passed
0247 1      to the action routine.
0248 1      !--
0249 1
0250 2 BEGIN
0251 2
0252 2 BUILTIN FP;
0253 2
0254 2 MAP
0255 2     msg_desc:  REF BLOCK [,BYTE];          ! Address of descriptor
0256 2
0257 2     .fp = nml$mainhandler;                  ! Enable condition handler
0258 2
0259 2     nml$gl_rcvdatlen = .msg_desc [dsc$w_length]; ! Copy length of message
0260 2
0261 2     CH$MOVE(.msg_desc [dsc$w_length],        ! Copy message itself
0262 2             .msg_desc [dsc$a_pointer],
0263 2             nml$ab_rcvbuffer);
0264 2
0265 2     nml$debug_msg(dbg$sc_netio,                ! Log type code
0266 2                   .msg_desc [dsc$a_pointer],    ! Message buffer address
0267 2                   .msg_desc [dsc$w_length],      ! Message data length
0268 2                   %ASCII 'NICE message received'); ! Header text
0269 2
0270 2     nml$gl_response_rtn = .resp_rtn;           ! Save address of response routine
0271 2
0272 2     IF NOT nml$parse_init()                    ! Parse received message
0273 2     THEN
0274 2         RETURN;                                ! Return on failure
0275 2
0276 2     IF nml$v2_compatibility()                  ! Process V2 NICE if necessary
0277 2     THEN
0278 2         RETURN;                                ! If it handled it, then exit
0279 2
0280 2     SELECTONEU .nml$gb_function                ! Dispatch the function
0281 2     OF
0282 2         SET
0283 2         [NMA$C_FNC_REA]:  NML$READ ();        ! Read
0284 2
```

```

288 0285 2 [NMASC_FNC_CHA]: NML$CHANGE (); ! Change
289 0286 2
290 0287 2 [NMASC_FNC_ZER]: NML$ZERO (); ! Zero
291 0288 2
292 0289 2 [NMASC_FNC_TES, ! Test
293 0290 2 NMASC_FNC_LOA, ! Load
294 0291 2 NMASC_FNC_TRI, ! Trigger
295 0292 2 NMASC_FNC_DUM]: NML$CALL_MOM (); ! Dump
296 0293 2
297 0294 2 [NMASC_FN2_LOO]: NML$LOOP2 (); ! Loop (Phase II)
298 0295 2
299 0296 2 [NMASC_FN2_REA, ! Read (Phase II SHOW)
300 0297 2 NMASC_FN2_ZER]: NML$PHASE2 (); ! Zero (Phase II)
301 0298 2
302 0299 2 [OTHERWISE]: NML$ERROR_1 (NMASC_STS_MPR);
303 0300 2 TES;
304 0301 1 END;
```

```

65 72 20 65 67 61 73 73 65 6D 20 45 43 49 4E 00000 P.AAB: .ASCII \NICE message received\<0><0><0>
00 00 00 64 65 76 69 65 63 0000F
010E0015 00018 P.AAA: .LONG 17694741
00000000 0001C .ADDRESS P.AAB
```

.PSECT \$SPLIT\$,NOWRT,NOEXE,2

```

00000000G 00 00000300G 04 6D 00000000V 00 007C 00000
56 04 AC D0 00009
00 66 3C 0000D
B6 66 28 00014
00000000 00 00000000 00 9F 0001D
7E 66 3C 00023
04 A6 DD 00026
00000000G 00 08 04 FB 0002B
00000000 00 AC D0 00032
00000000G 00 00 FB 0003A
73 50 E9 00041
00000000G 00 00 FB 00044
69 50 E8 0004B
52 00000000G 00 9A 0004E
14 52 91 00055
08 12 00058
00000000G 00 00 FB 0005A
04 00061
13 52 91 00062 1$:
08 12 00065
00000000G 00 00 FB 00067
04 0006E
15 52 91 0006F 2$:
08 12 00072
```

.PSECT \$CODE\$,NOWRT,2

```

.ENTRY NML$PROCESS_NICE, Save R2,R3,R4,R5,R6 : 0230
MOVAB NML$MAINHANDLER, (FP) : 0257
MOVL MSG_DESC, R6 : 0259
MOVZWL (R6), NML$GL_RCVDATLEN
MOVCL (R6), @4(R6), NML$AB_RCVBUFFER : 0261
PUSHAB P.AAA : 0267
MOVZWL (R6), -(SP)
PUSHL 4(R6) : 0266
CLRL -(SP) : 0265
CALLS #4, NML$DEBUG_MSG
MOVL RESP_RTN, NML$GL_RESPONSE_RTN : 0270
CALLS #0, NML$PARSE_INIT : 0272
BLBC R0, 7$ : 0276
CALLS #0, NML$V2_COMPATIBILITY
BLBS R0, 7$ : 0280
MOVZBL NML$GB_FUNCTION, R2 : 0283
CMPB R2, #20
BNEQ 1$
CALLS #0, NML$READ
RET
CMPB R2, #19 : 0285
BNEQ 2$
CALLS #0, NML$CHANGE
RET
CMPB R2, #21 : 0287
BNEQ 3$
```

NML\$ENTRY  
V04-000

Network Management Listener entry point  
NML\$PROCESS\_NICE Main command processing routine

B 10  
15-Sep-1984 23:58:02  
14-Sep-1984 12:50:08

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[NML.SRC]NMLENTRY.B32;1  
Page 11  
(4)

00000000G	00	00	FB	00074	CALLS	#0, NML\$ZERO	:	
	0F	52	04	0007B	RET		:	
		0D	91	0007C	3\$: CMPB	R2, #15	:	0289
	12	52	1F	0007F	BLSSU	4\$	:	
		08	91	00081	CMPB	R2, #18	:	
00000000G	00	00	1A	00084	BGTRU	4\$	:	
		00	FB	00086	CALLS	#0, NML\$CALL_MOM	:	0292
	05	52	04	0008D	RET		:	
		08	91	0008E	4\$: CMPB	R2, #5	:	0294
		08	12	00091	BNEQ	5\$	:	
00000000V	00	00	FB	00093	CALLS	#0, NML\$LOOP2	:	
		00	04	0009A	RET		:	
	08	52	91	0009B	5\$: CMPB	R2, #8	:	0296
		0D	1F	0009E	BLSSU	6\$	:	
	09	52	91	000A0	CMPB	R2, #9	:	
		08	1A	000A3	BGTRU	6\$	:	
00000000V	00	00	FB	000A5	CALLS	#0, NML\$PHASE2	:	0297
		00	04	000AC	RET		:	
	7E	05	CE	000AD	6\$: MNEGL	#5, -(SP)	:	0299
00000000G	00	01	FB	000B0	CALLS	#1, NML\$ERROR_1	:	
		04	000B7	7\$: RET			:	0301

; Routine Size: 184 bytes, Routine Base: \$CODE\$ + 0074

NM  
VO

```

0306 1 %SBTTL 'NML$TERMINATE Terminate NICE communications'
0307 1
0308 1 GLOBAL ROUTINE NML$TERMINATE: NOVALUE =
0309 1
0310 1 ++
0311 1
0312 1 This routine is called to terminate communications with this
0313 1 listener. It cleans up any database or storage if needed.
0314 1
0315 1 Inputs:
0316 1
0317 1 None
0318 1
0319 1 Outputs:
0320 1
0321 1 None - all errors are signaled.
0322 1
0323 1 --
0324 1 BEGIN
0325 2
0326 2 nml$closefile (NML$C_OPEN_ALL); ! Close any open files
0327 2
0328 1 END;

```

```

0000 0000
7E 7F 8F 9A 00002
00000000G 00 01 FB 00006
04 0000D

```

```

.ENTRY NML$TERMINATE, Save nothing
MOVZBL #127, -(SP)
CALLS #1, NML$CLOSEFILE
RET

```

```

: 0304
: 0322
: 0324

```

; Routine Size: 14 bytes, Routine Base: \$CODE\$ + 012C

```
0325 1 %SBTTL 'NML_INITLOG Initialization debug logging'
0326 1
0327 1 ROUTINE NML_INITLOG: NOVALUE =
0328 1
0329 1 ++
0330 1
0331 1 This routine initializes the internal logging flags for NML debugging.
0332 1 The logical name NML$LOG is translated to get the flag settings.
0333 1 Also, if the logical name NML$WATCHER translates, log all NICE
0334 1 messages received and sent by NML. Useful for keeping a running log
0335 1 of all network management changes done on a node for as long as
0336 1 NML$WATCHER is defined.
0337 1
0338 1 Inputs:
0339 1
0340 1 None
0341 1
0342 1 Outputs:
0343 1
0344 1 None
0345 1 --
0346 1
0347 2 BEGIN
0348 2
0349 2 |
0350 2 | Set internal logging flags if NML$LOG is defined.
0351 2 |
0352 2 |
0353 2 NML$TRNLOGNUM ($ASCII ('NML$LOG'), NML$GL_LOGMASK);
0354 2
0355 2 |
0356 2 | If the NPARSE logging flag is set then set it in the NPARSE data area.
0357 2 |
0358 2 |
0359 2 IF .NML$GL_LOGMASK [DBG$C_NPARSE]
0360 2 THEN
0361 2 NPA$GL_LOGMASK = 1
0362 2 ELSE
0363 2 NPA$GL_LOGMASK = 0;
0364 2
0365 2 |
0366 2 | Log contents of permanent data base files.
0367 2 |
0368 2 |
0369 2 NML$LOGALLPDB ();
0370 2
0371 2 |
0372 2 |
0373 2 | If the logical name NML$WATCHER translates, log all NICE
0374 2 | messages received and sent by NML. Useful for keeping a running log
0375 2 | of all network management changes done on a node for as long as
0376 2 | NML$WATCHER is defined.
0377 2 |
0378 2 |
0379 2 |
0380 2 |
0381 2 |
0382 2 |
0383 2 P $ASSIGN (DEVNAM = NML$GQ_WATCHER DSC,
0384 2 CHAN = NML$GW_WATCHER_CHAN);
0385 2 END;
```

```

                                .PSECT $SPLITS$,NOWRT,NOEXE,2
                                .ASCII \NML$LOG\
                                .BLKB 1
                                .LONG 7
                                .ADDRESS P.AAD
                                .EXTRN SYSS$ASSIGN
                                .PSECT $CODES$,NOWRT,2

                                000C 00000 NML_INITLOG:
                                .WORD Save R2,R3
                                MOVAB NML$GL_LOGMASK, R3
                                MOVAB NPASGL_LOGMASK, R2
                                PUSHL R3
                                PUSHAB P.AAC
                                CALLS #2, NML$STRNLOGNUM
                                BBC #2, NML$GL_LOGMASK, 1$
                                MOVL #1, NPASGL_LOGMASK
                                BRB 2$
                                CLRL NPASGL_LOGMASK
                                CALLS #0, NML$LOGALLPDB
                                CLRB -(SP)
                                PUSHAB NML$GW_WATCHER_CHAN
                                PUSHAB NML$GQ_WATCHER_DSC
                                CALLS #4, SYSS$ASSIGN
                                RET

```

0327  
0353  
0359  
0361  
0363  
0369  
0379  
0380

; Routine Size: 71 bytes, Routine Base: \$CODES + 013A

```

387 0381 1 %SBTTL 'NML$SEND Send NICE response to caller'
388 0382 1
389 0383 1 GLOBAL ROUTINE NML$SEND (BUFADR, BUFLen) =
390 0384 1
391 0385 1 !++
392 0386 1
393 0387 1 This routine sends NICE protocol status messages back
394 0388 1 to the NICE caller.
395 0389 1
396 0390 1 Inputs:
397 0391 1
398 0392 1 bufadr Address of the buffer to be transmitted.
399 0393 1 buflen Length of the buffer in bytes.
400 0394 1
401 0395 1 nml$gl_response_rtn Channel assigned to the command process link.
402 0396 1
403 0397 1 Outputs:
404 0398 1
405 0399 1 Returns success. Errors are signalled.
406 0400 1 !--
407 0401 1
408 0402 2 BEGIN
409 0403 2
410 0404 2 LOCAL
411 0405 2 desc: VECTOR [2]; ! Descriptor of response message
412 0406 2
413 0407 2 nml$debug_msg(dbg$C_netio, ! Log message transmitted
414 0408 2 .bufadr,
415 0409 2 .buflen,
416 0410 2 %ASCII 'NICE message transmitted');
417 0411 2
418 0412 2 desc [0] = .buflen; ! Setup descriptor of response
419 0413 2 desc [1] = .bufadr;
420 0414 2
421 0415 2 (.nml$gl_response_rtn) (desc); ! Call caller's response action routine
422 0416 2
423 0417 2 RETURN true; ! Return successful
424 0418 2
425 0419 1 END;

```

```

72 74 20 65 67 61 73 73 65 6D 20 45 43 49 4E 00030 P.AAF: .ASCII \NICE message transmitted\
64 65 74 74 69 6D 73 6E 61 0003F
010E0018 00048 P.AAE: .LONG 17694744
00000000' 0004C .ADDRESS P.AAF

```

```

.PSECT $CODE$,NOWRT,2

SE 00000000' 0000 00000 .ENTRY NML$SEND, Save nothing
08 C2 00002
00 9F 00005
7E 04 AC 7D 0000B .SUBL2 #8, SP
PUSHAB P.AAE
MOVQ BUFADR, -(SP)

```

```

: 0383
: 0409
: 0408

```

NML\$ENTRY  
V04-000

Network Management Listener entry point  
NML\$SEND Send NICE response to caller

6 10  
15-Sep-1984 23:58:02  
14-Sep-1984 12:50:08

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[NML.SRC]NMLENTRY.B32;1

Page 16  
(7)

```
00000000G 00      7E D4 0000F
              04 FB 00011
              08 AC D0 00018
04 AE      04 AC D0 0001C
50 00000000' 00 D0 00021
              SE DD 00028
              01 FB 0002A
              01 D0 0002D
              04 00030
```

```
CLRL -(SP)
CALLS #4, NML$DEBUG_MSG
MOVL BUFLN, DESC
MOVL BUFADR, DESC+4
MOVL NML$GL_RESPONSE_RTN, R0
PUSHL SP
CALLS #1, (R0)
MOVL #1, R0
RET
```

```
: 0407
:
: 0412
: 0413
: 0415
:
: 0417
: 0419
```

: Routine Size: 49 bytes. Routine Base: \$CODE\$ + 0181

```
427 0420 1 %SBTTL 'NML$LOOP2 Phase II passive loopback'
428 0421 1
429 0422 1 ROUTINE NML$LOOP2 : NOVALUE =
430 0423 1
431 0424 1 ++
432 0425 1 FUNCTIONAL DESCRIPTION:
433 0426 1
434 0427 1 This routine acts as the phase II loopback mirror.
435 0428 1
436 0429 1 FORMAL PARAMETERS:
437 0430 1
438 0431 1 NONE
439 0432 1
440 0433 1 IMPLICIT INPUTS:
441 0434 1
442 0435 1 NML$AB_RCVBUFFER contains the received message.
443 0436 1 NML$GL_RCVDATLEN contains the length of the received data.
444 0437 1
445 0438 1 IMPLICIT OUTPUTS:
446 0439 1
447 0440 1 NML$AB_RCVBUFFER is altered.
448 0441 1
449 0442 1 ROUTINE VALUE:
450 0443 1 COMPLETION CODE:
451 0444 1
452 0445 1 NONE
453 0446 1
454 0447 1 SIDE EFFECTS:
455 0448 1
456 0449 1 Signals response message.
457 0450 1
458 0451 1 --
459 0452 1
460 0453 1 BEGIN
461 0454 1
462 0455 1 Make sure that it is a valid loopback message.
463 0456 1 If it is valid then set message header to 1 and send message
464 0457 1 else set message header to -1 and send message.
465 0458 1
466 0459 1 IF .(NML$AB_RCVBUFFER + 1)<0,8,0> EQL 0
467 0460 1 THEN
468 0461 1 BEGIN
469 0462 1
470 0463 1 (NML$AB_RCVBUFFER + 1)<0,8,0> = 1;
471 0464 1 $SIGNAL_MSG (NML$AB_RCVBUFFER + 1, .NML$GL_RCVDATLEN - 1);
472 0465 1
473 0466 1 END
474 0467 1 ELSE
475 0468 1 BEGIN
476 0469 1
477 0470 1 (NML$AB_RCVBUFFER + 1)<0,8,0> = -1;
478 0471 1 $SIGNAL_MSG (NML$AB_RCVBUFFER + 1, 1);
479 0472 1
480 0473 1 END;
481 0474 1
482 0475 1 END; ! End of NML$LOOP2
```

```

                                0004 00000 NML$LOOP2:
                                .WORD      Save R2
                                MOVAB      NML$AB_RCVBUFFER+1, R2
                                TSTB       NML$AB_RCVBUFFER+1
                                BNEQ       1$
                                MOVB       #1, NML$AB_RCVBUFFER+1
                                SUBL3      #1, NML$GL_RCVDATLEN, -(SP)
                                BRB        2$
                                MNEGB      #1, NML$AB_RCVBUFFER+1
                                PUSHL      #1
                                PUSHL      R2
                                PUSHL      #33095680
                                CALLS      #3, LIB$SIGNAL
                                RET
                                : 0422
                                : 0459
                                : 0463
                                : 0464
                                : 0470
                                : 0471
                                : 0475

```

; Routine Size: 47 bytes, Routine Base: \$CODE\$ + 01B2

```
484 0476 1 %SBTTL 'NML$PHASE2 Routine which connects to NICE'
485 0477 1
486 0478 1 ROUTINE NML$PHASE2 : NOVALUE =
487 0479 1
488 0480 1
489 0481 1 ++
490 0482 1 FUNCTIONAL DESCRIPTION:
491 0483 1 This routine passes PHASE2 commands to the NICE object and
492 0484 1 returns to the command process, the responses from the NICE object
493 0485 1
494 0486 1 FORMAL PARAMETERS:
495 0487 1
496 0488 1 NONE
497 0489 1
498 0490 1 IMPLICIT INPUTS:
499 0491 1
500 0492 1 NML$W_NICECHAN NICE object channel.
501 0493 1
502 0494 1 ROUTINE VALUE:
503 0495 1 COMPLETION CODE:
504 0496 1
505 0497 1 All errors are signalled. Otherwise the value NML$STS_SUC is
506 0498 1 returned.
507 0499 1
508 0500 1 SIDE EFFECTS:
509 0501 1
510 0502 1 NONE
511 0503 1 --
512 0504 1
513 0505 1 BEGIN
514 0506 1
515 0507 1 LITERAL
516 0508 1 SNDBUFSIZE = 256;
517 0509 1
518 0510 1 LOCAL
519 0511 1 COUNT : WORD, ! Contains number of data messages
520 0512 1 ! received from NICE task
521 0513 1
522 0514 1 STATUS,
523 0515 1 RCV_IOSB : $IOSB,
524 0516 1 XMIT_IOSB : $IOSB;
525 0517 1
526 0518 1 Connect information for NICE object for Phase 2 processing.
527 0519 1
528 0520 1 BIND
529 0521 1 NICEOBJECTDSC = $ASCID (':':"TASK=NMLPH2"' ) : DESCRIPTOR;
530 0522 1
531 0523 1 If Phase 2 command process then attempt to connect to NICE object.
532 0524 1
533 0525 1 IF .NML$B_PH2LINK
534 0526 1 THEN
535 0527 1 BEGIN
536 0528 1 STATUS = $ASSIGN (CHAN = NML$W_NICECHAN,
537 0529 1 DEVNAM = NICEOBJECTDSC);
538 0530 1 IF NOT .STATUS
539 0531 1 THEN
540 0532 1 NML$ERROR_1 (NML$C_STS_RES);
```

```
541 0533
542 0534
543 0535
544 0536
545 0537
546 0538
547 0539
548 0540
549 0541
550 0542
551 0543
552 0544
553 0545
554 0546
555 0547
556 0548
557 0549
558 0550
559 0551
560 0552
561 0553
562 0554
563 0555
564 0556
565 0557
566 0558
567 0559
568 0560
569 0561
570 0562
571 0563
572 0564
573 0565
574 0566
575 0567
576 0568
577 0569
578 0570
579 0571
580 0572
581 0573
582 0574
583 0575
584 0576
585 0577
586 0578
587 0579
588 0580
589 0581
590 0582
591 0583
592 0584
593 0585
594 0586
595 0587
596 0588
597 0589

END;

Attempt to transmit Phase II command to NICE.

STATUS = $QIOW (CHAN = .NML$W_NICECHAN,
                FUNC = IOS_WRITEVBLK,
                IOSB = XMIT_IOSB,
                P1 = NML$AB_RCVBUFFER,
                P2 = .NML$GC_RCVDATLEN);

IF .STATUS
THEN
    STATUS = .XMIT_IOSB [IOS$W_STATUS];

IF NOT .STATUS
THEN
    NML$ERROR_1 (NMASC_STS_RES);

If transmit was successful then post read to NICE

STATUS = $QIOW (CHAN = .NML$W_NICECHAN,
                FUNC = IOS_READVBLK,
                IOSB = RCV_IOSB,
                P1 = NML$AB_SNDBUFFER,
                P2 = SNDBUFSIZE);

IF .STATUS
THEN
    STATUS = .RCV_IOSB [IOS$W_STATUS];

IF NOT .STATUS
THEN
    NML$ERROR_1 (NMASC_STS_RES);

If receive was successful then send received NICE message
to requestor of command.

STATUS = NML$SEND (NML$AB_SNDBUFFER,
                  .RCV_IOSB [IOS$W_COUNT]);

If send was successful then continue reading data messages

IF NOT .STATUS
THEN
    NML$ERROR_1 (NMASC_STS_RES);

IF .RCV_IOSB [IOS$W_COUNT] LESS 3
THEN
    COUNT = 0
ELSE
    COUNT = .(NML$AB_SNDBUFFER+1)<0,16,0>;

DECR I FROM .COUNT-1 TO 0 DO
    BEGIN
        STATUS = $QIOW (CHAN = .NML$W_NICECHAN,
```

```
598      P 0590      FUNC = IOS_READVBLK,  
599      P 0591      IOSB = RCV_IOSB,  
600      P 0592      P1 = NML$AB_SNDBUFFER,  
601      P 0593      P2 = SNDBUF5IZE);  
602      P 0594  
603      P 0595      IF .STATUS  
604      P 0596      THEN  
605      P 0597          STATUS = .RCV_IOSB[IOS$W_STATUS];  
606      P 0598  
607      P 0599      IF NOT .STATUS  
608      P 0600      THEN  
609      P 0601          NML$ERROR_1(NMASC_STS_RES);  
610      P 0602  
611      P 0603      STATUS = NML$SEND (NML$AB_SNDBUFFER,  
612      P 0604          .RCV_IOSB [IOS$W_COUNT]);  
613      P 0605  
614      P 0606      IF NOT .STATUS  
615      P 0607      THEN  
616      P 0608          NML$ERROR_1 (NMASC_STS_RES);  
617      P 0609  
618      P 0610      END;                                ! End of DECR block  
619      P 0611  
620      P 0612  
621      P 0613      RETURN NML$_STS_SUC;  
622      P 0614  
623      P 0615      END;                                ! End of NML$PHASE2
```

```
22 32 48 50 4C 4D 4E 3D 4B 53 41 54 22 3A 3A 00050 P.AAH: .ASCII \::"TASK=NMLPH2"\ :  
0005F .BLKB 1 :  
0000000F 00060 P.AAG: .LONG 15 :  
00000000 00064 .ADDRESS P.AAH :  
  
NICEOBJECTDSC= P.AAG  
.EXTRN SYSSQIOW  
  
.PSECT $CODE$,NOWRT,2  
  
01FC 0000 NML$PHASE2:  
58      9B AF 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7,R8 : 0478  
57 00000000G 00 9E 00006 MOVAB NML$SEND, R8  
56 00000000G 00 9E 0000D MOVAB SYSSQIOW, R7  
55 00000000G 00 9E 00014 MOVAB NML$W_NICECHAN, R6  
54 00000000G 00 9E 0001B MOVAB NML$AB_SNDBUFFER, R5  
5E 10 C2 00022 MOVAB NML$ERROR_1, R4  
1D FE A6 E9 00025 SUBL2 #16, SP  
7E 7C 00029 BLBC NML$B_PH2LINK, 1$ : 0524  
56 DD 0002B CLRQ -(SP) : 0529  
00000000G 00 9F 0002D PUSHL R6  
00 04 FB 00033 PUSHAB NICEOBJECTDSC  
52 50 D0 0003A CALLS #4, SYSSASSIGN  
06 52 E8 0003D MOVL R0, STATUS : 0530  
7E 0F CE 00040 BLBS STATUS, 1$ : 0532  
MNEGL #15, -(SP)
```

64		01	FB	00043	CALLS	#1, NML\$ERROR_1	
		7E	7C	00046	1\$: CLRQ	-(SP)	0542
		7E	7C	00048	CLRQ	-(SP)	
	00000000G	00	DD	0004A	PUSHL	NML\$GL_RCVDATLEN	
	00000000G	00	9F	00050	PUSHAB	NML\$AB_RCVBUFFER	
		7E	7C	00056	CLRQ	-(SP)	
	20	AE	9F	00058	PUSHAB	XMIT_IOSB	
		30	DD	0005B	PUSHL	#48	
7E		66	3C	0005D	MOVZWL	NML\$W_NICECHAN, -(SP)	
		7E	D4	00060	CLRL	-(SP)	
67		0C	FB	00062	CALLS	#12, SYSSQIOW	
52		50	DD	00065	MOVL	R0, STATUS	
06		52	E9	00068	BLBC	STATUS, 2\$	0544
52		6E	3C	0006B	MOVZWL	XMIT_IOSB, STATUS	0546
06		52	E8	0006E	BLBS	STATUS, 3\$	0548
7E		0F	CE	00071	2\$: MNEGL	#15, -(SP)	0550
64		01	FB	00074	CALLS	#1, NML\$ERROR_1	
		7E	7C	00077	3\$: CLRQ	-(SP)	0558
		7E	7C	00079	CLRQ	-(SP)	
7E	0100	8F	3C	0007B	MOVZWL	#256, -(SP)	
		55	DD	00080	PUSHL	R5	
		7E	7C	00082	CLRQ	-(SP)	
	28	AE	9F	00084	PUSHAB	RCV_IOSB	
		31	DD	00087	PUSHL	#49	
7E		66	3C	00089	MOVZWL	NML\$W_NICECHAN, -(SP)	
		7E	D4	0008C	CLRL	-(SP)	
67		0C	FB	0008E	CALLS	#12, SYSSQIOW	
52		50	DD	00091	MOVL	R0, STATUS	
07		52	E9	00094	BLBC	STATUS, 4\$	0560
52	08	AE	3C	00097	MOVZWL	RCV_IOSB, STATUS	0562
06		52	E8	0009B	BLBS	STATUS, 5\$	0564
7E		0F	CE	0009E	4\$: MNEGL	#15, -(SP)	0566
64		01	FB	000A1	CALLS	#1, NML\$ERROR_1	
7E	0A	AE	3C	000A4	5\$: MOVZWL	RCV_IOSB+2, -(SP)	0572
		55	DD	000A8	PUSHL	R5	0571
68		02	FB	000AA	CALLS	#2, NML\$SEND	
52		50	DD	000AD	MOVL	R0, STATUS	
06		52	E8	000B0	BLBS	STATUS, 6\$	0576
7E		0F	CE	000B3	MNEGL	#15, -(SP)	0578
64		01	FB	000B6	CALLS	#1, NML\$ERROR_1	
03	0A	AE	B1	000B9	6\$: CMPW	RCV_IOSB+2, #3	0580
		04	1E	000BD	BGEQU	7\$	
		50	B4	000BF	CLRW	COUNT	0582
		04	11	000C1	BRB	8\$	
50	01	A5	B0	000C3	7\$: MOVW	NML\$AB_SNDBUFFER+1, COUNT	0584
53		50	3C	000C7	8\$: MOVZWL	COUNT, -1	0586
		42	11	000CA	BRB	12\$	
		7E	7C	000CC	9\$: CLRQ	-(SP)	0593
		7E	7C	000CE	CLRQ	-(SP)	
7E	0100	8F	3C	000D0	MOVZWL	#256, -(SP)	
		55	DD	000D5	PUSHL	R5	
		7E	7C	000D7	CLRQ	-(SP)	
	28	AE	9F	000D9	PUSHAB	RCV_IOSB	
		31	DD	000DC	PUSHL	#49	
7E		66	3C	000DE	MOVZWL	NML\$W_NICECHAN, -(SP)	
		7E	D4	000E1	CLRL	-(SP)	
67		0C	FB	000E3	CALLS	#12, SYSSQIOW	

NML\$ENTRY  
V04-000

Network Management Listener entry point  
NML\$PHASE2 Routine which connects to NICE

N 10  
15-Sep-1984 23:58:02  
14-Sep-1984 12:50:08

VAX-11 Bliss-32 V4.0-742  
DISK\$VMSMASTER:[NML.SRC]NMLENTRY.B32;1 Page 23  
(9)

52		50	D0	000E6	MOVL	R0, STATUS		
07		52	E9	000E9	BLBC	STATUS, 10\$	...	0595
52	08	AE	3C	000EC	MOVZWL	RCV_IOSB, STATUS	...	0597
06		52	E8	000F0	BLBS	STATUS, 11\$	...	0599
7E		0F	CE	000F3	MNEGL	#15, -(SP)	...	0601
64		01	FB	000F6	CALLS	#1, NML\$ERROR_1	...	
7E	0A	AE	3C	000F9	MOVZWL	RCV_IOSB+2, -(SP)	...	0604
		55	DD	000FD	PUSHL	R5	...	0603
68		02	FB	000FF	CALLS	#2, NML\$SEND	...	
52		50	D0	00102	MOVL	R0, STATUS	...	
06		52	E8	00105	BLBS	STATUS, 12\$	...	0606
7E		0F	CE	00108	MNEGL	#15, -(SP)	...	0608
64		01	FB	0010B	CALLS	#1, NML\$ERROR_1	...	
BB		53	F4	0010E	SOBGEQ	I, 9\$	...	0586
		04	00111	RET			...	0615

; Routine Size: 274 bytes. Routine Base: \$CODE\$ + 01E1

```

625 0616 1 XSBTTL 'NML$MAINHANDLER Condition handler routine'
626 0617 1
627 0618 1 GLOBAL ROUTINE NML$MAINHANDLER (SIGNAL_VEC, MECHANISM) =
628 0619 1
629 0620 1 ++
630 0621 1 FUNCTIONAL DESCRIPTION:
631 0622 1
632 0623 1     This is the condition handler routine for NML.
633 0624 1
634 0625 1 FORMAL PARAMETERS:
635 0626 1
636 0627 1     SIGNAL_VEC      Signal vector block.
637 0628 1     MECHANISM        Mechanism vector argument block.
638 0629 1
639 0630 1 IMPLICIT INFUTS:
640 0631 1
641 0632 1     NONE
642 0633 1
643 0634 1 IMPLICIT OUTPUTS:
644 0635 1
645 0636 1     NONE
646 0637 1
647 0638 1 ROUTINE VALUE:
648 0639 1 COMPLETION CODES:
649 0640 1
650 0641 1     NONE
651 0642 1
652 0643 1 SIDE EFFECTS:
653 0644 1
654 0645 1     NONE
655 0646 1
656 0647 1 --
657 0648 1
658 0649 1 BEGIN
659 0650 1
660 0651 1 MAP
661 0652 1     SIGNAL_VEC : REF BBLOCK,      ! Signal vector arg
662 0653 1     MECHANISM : REF BBLOCK;    ! Mechanism vector arg
663 0654 1
664 0655 1 LOCAL
665 0656 1     BUF_ADR,      ! Temporary buffer address
666 0657 1     BUF_LEN,      ! Temporary buffer length
667 0658 1     STS_CODE : BBLOCK [4]; ! Status code
668 0659 1
669 0660 1     STS_CODE = .SIGNAL_VEC [CHF$SIG_NAME]; ! Get signal status code
670 0661 1
671 0662 1 Facility code must match the one for NML.
672 0663 1
673 0664 1 IF .STS_CODE [ST$V_FAC_NO] EQLU NML$K_FAC_CODE
674 0665 1 THEN
675 0666 1     BEGIN
676 0667 1
677 0668 1 Two arguments are required for NML conditions.
678 0669 1
679 0670 1 IF .SIGNAL_VEC [CHF$SIG_ARGS] NEQU 2+3
680 0671 1 THEN
681 0672 1     RETURN SS$_RESIGNAL
```

```

: 682      0673 3      ELSE
: 683      0674 4      BEGIN
: 684      0675 4
: 685      0676 4      BUF_ADR = .SIGNAL_VEC [CHF$SIG_ARG1];
: 686      0677 4      BUF_LEN = .(SIGNAL_VEC [CHF$SIG_ARG1]+4);
: 687      0678 4
: 688      0679 4      If a message is specified (length not equal 0) then send it.
: 689      0680 4
: 690      0681 4      IF .BUF_LEN NEQU 0
: 691      0682 4      THEN
: 692      0683 4          NML$SEND (.BUF_ADR, .BUF_LEN); ! Send status message
: 693      0684 4
: 694      0685 4          MECHANISM [CHF$MCH_SAVRO] = 0;
: 695      0686 4
: 696      0687 4      Unwind back to the routine that set up the condition handler and continue
: 697      0688 4      from there.
: 698      0689 4
: 699      0690 4          $UNWIND (DEPADR = MECHANISM [CHF$MCH_DEPTH]);
: 700      0691 4          RETURN SS$_CONTINUE
: 701      0692 4
: 702      0693 3      END;
: 703      0694 3      ELSE
: 704      0695 2
: 705      0696 2      This condition was not signalled by NML so let it go by.
: 706      0697 2
: 707      0698 2
: 708      0699 2      RETURN SS$_RESIGNAL
: 709      0700 2
: 710      0701 1      END;

```

! End of NML\$MAINHANDLER

000001F9 8F

51

FE61

00000000G

```

50      04      AC D0 00002
51      04      A0 D0 00006
0C      10      ED 0000A
      2F      12 00013
05      60      D1 00015
      2A      12 00018
51      08      A0 D0 0001A
50      0C      A0 D0 0001E
      09      13 00022
      50      DD 00024
      51      DD 00026
FE61    CF      02      FB 00028
50      08      AC D0 0002D 1$:
      0C      A0 D4 00031
      7E      D4 00034
      08      A0 9F 00036
00000000G 00      02      FB 00039
50      01      D0 00040
      04      00043
50      0918    8F      3C 00044 2$:
      04      00049

```

.EXTRN SYSSUNWIND

```

.ENTRY NML$MAINHANDLER, Save nothing
MOVL   SIGNAL_VEC, R0
MOVL   4(R0), STS_CODE
CMPZV  #16, #12, STS_CODE, #505
BNEQ   2$
CMPL   (R0), #5
BNEQ   2$
MOVL   8(R0), BUF_ADR
MOVL   12(R0), BUF_LEN
BEQL   1$
PUSHL  BUF_LEN
PUSHL  BUF_ADR
CALLS  #2, NML$SEND
MOVL   MECHANISM, R0
CLRL   12(R0)
CLRL   -(SP)
PUSHAB 8(R0)
CALLS  #2, SYSSUNWIND
MOVL   #1, R0
RET
MOVZWL #2328, R0
RET

```

```

: 0618
: 0660
: 0664
: 0670
: 0676
: 0677
: 0681
: 0683
: 0685
: 0690
: 0691
: 0699
: 0701

```

NML\$ENTRY  
V04-000

Network Management Listener entry point  
NML\$MAINHANDLER Condition handler routine

; Routine Size: 74 bytes, Routine Base: \$CODE\$ + 02F3

; 711 0702 1

D 11  
15-Sep-1984 23:58:02  
14-Sep-1984 12:50:08

VAX-11 BLISS-32 V4.0-742  
DISK\$VMSMASTER:[NML.SRC]NMLENTRY.B32;1 (10)

Page 26

.EXTRN LIB\$SIGNAL

PSECT SUMMARY

Name	Bytes	Attributes					
\$OWNS	36	NOVEC,	WRT,	RD	,NOEXE,NOSHR,	LCL,	REL, CON,NOPIC,ALIGN(2)
\$CODE\$	829	NOVEC,NOWRT,	RD	, EXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)
\$PLITS	104	NOVEC,NOWRT,	RD	,NOEXE,NOSHR,	LCL,	REL,	CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
..\$255\$DUA28:[NML.OBJ]NMLLIB.L32;1	341	32	9	27	00:00.1
..\$255\$DUA28:[SHRLIB]NMLIBRY.L32;1	887	13	1	47	00:00.2
..\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	21	0	581	00:02.2

COMMAND QUALIFIERS

```

:
:      BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:NMLENTRY/OBJ=OBJ$:NMLENTRY MSRC$:NMLENTRY/UPDATE=(ENH$:NMLENTRY)
:
: Size:      829 code + 140 data bytes
: Run Time:   00:17.9
: Elapsed Time: 00:42.8
: Lines/CPU Min: 2361
: Lexemes/CPU-Min: 14149
: Memory Used: 137 pages
: Compilation Complete

```

0283 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

